

FFFFFFFFFFFFFFFF	111	111	AAAAAAAAA	
FFFFFFFFFFFFFFFF	111	111	AAAAAAAAA	
FFFFFFFFFFFFFFFF	111	111	AAAAAAAAA	
FFF	111111	111111	AAA	AAA
FFF	111111	111111	AAA	AAA
FFF	111111	111111	AAA	AAA
FFF	111	111	AAA	AAA
FFF	111	111	AAA	AAA
FFF	111	111	AAA	AAA
FFFFFFFFFFFFFF	111	111	AAA	AAA
FFFFFFFFFFFFFF	111	111	AAA	AAA
FFFFFFFFFFFFFF	111	111	AAA	AAA
FFF	111	111	AAAAAAAAAAAAAAAA	
FFF	111	111	AAAAAAAAAAAAAAAA	
FFF	111	111	AAAAAAAAAAAAAAAA	
FFF	111	111	AAA	AAA
FFF	111	111	AAA	AAA
FFF	111	111	AAA	AAA
FFF	111	111	AAA	AAA
FFF	111111111	111111111	AAA	AAA
FFF	111111111	111111111	AAA	AAA
FFF	111111111	111111111	AAA	AAA

```
PPPPPPPP  MM      MM  SSSSSSSS
PPPPPPPP  MM      MM  SSSSSSSS
PP        PP  MMMM  MMMM  SS
PP        PP  MMMM  MMMM  SS
PP        PP  MM   MM   MM  SS
PP        PP  MM   MM   MM  SS
PPPPPPPP  MM      MM  SSSSSS
PPPPPPPP  MM      MM  SSSSSS
PP        MM      MM      SS
PP        MM      MM      SS
PP        MM      MM      SS
PP        MM      MM      SS
PP        MM      MM  SSSSSSSS
PP        MM      MM  SSSSSSSS
```

```
....
....
....
....
```

```
LL        IIIIII  SSSSSSSS
LL        IIIIII  SSSSSSSS
LL        II      SS
LL        II      SS
LL        II      SS
LL        II      SS
LL        II      SSSSSS
LL        II      SSSSSS
LL        II      SS
LL        II      SS
LL        II      SS
LL        II      SS
LLLLLLLLLL IIIIII  SSSSSSSS
LLLLLLLLLL IIIIII  SSSSSSSS
```



```
1 0001 0 MODULE PMS (  
2 0002 0  
3 0003 0 LANGUAGE (BLISS32),  
4 0004 0 IDENT = 'V04-000'  
5 0005 1 BEGIN  
6 0006 1  
7 0007 1  
8 0008 1  
9 0009 1 *****  
10 0010 1 *  
11 0011 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY  
12 0012 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.  
13 0013 1 * ALL RIGHTS RESERVED.  
14 0014 1 *  
15 0015 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED  
16 0016 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE  
17 0017 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER  
18 0018 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY  
19 0019 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY  
20 0020 1 * TRANSFERRED.  
21 0021 1 *  
22 0022 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE  
23 0023 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT  
24 0024 1 * CORPORATION.  
25 0025 1 *  
26 0026 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS  
27 0027 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.  
28 0028 1 *  
29 0029 1 *****  
30 0030 1  
31 0031 1 ++  
32 0032 1  
33 0033 1 FACILITY: F11ACP Structure Level 1  
34 0034 1  
35 0035 1 ABSTRACT:  
36 0036 1  
37 0037 1 This module contains the local performance measurement data base  
38 0038 1 and the performance metering routines.  
39 0039 1  
40 0040 1 ENVIRONMENT:  
41 0041 1  
42 0042 1 STARLET operating system, including privileged system services  
43 0043 1 and internal exec routines.  
44 0044 1  
45 0045 1 --  
46 0046 1  
47 0047 1  
48 0048 1 AUTHOR: Andrew C. Goldstein, CREATION DATE: 25-Aug-1977 11:30  
49 0049 1  
50 0050 1 MODIFIED BY:  
51 0051 1  
52 0052 1 A0100 ACG0001 Andrew C. Goldstein, 10-Oct-1978 20:01  
53 0053 1 Previous revision history moved to F11A.REV  
54 0054 1  
55 0055 1 **  
56 0056 1  
57 0057 1
```

PMS
V04-000

J 6
16-Sep-1984 01:12:50
14-Sep-1984 12:29:47

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[F11A.SRC]PMS.B32;1

Page 2
(1)

```
: 58      0058 1 LIBRARY 'SYSS$LIBRARY:LIB,L32';
: 59      0059 1 REQUIRE 'SRC$:FCPDEF.B32';
: 60      0374 1
: 61      0375 1
: 62      0376 1 FORWARD ROUTINE
: 63      0377 1 PMS_START      : NOVALUE,      ! start measuring main function
: 64      0378 1 PMS_END        : NOVALUE,      ! end measuring main function
: 65      0379 1 PMS_START_SUB  : NOVALUE,      ! start measuring subfunction
: 66      0380 1 PMS_END_SOB    : NOVALUE;      ! end measuring subfunction
```



```

68 0381 1  +
69 0382 1
70 0383 1
71 0384 1 The performance measurement data base consists of the accounting array in
72 0385 1 system space, as well as some local storage to keep intermediate figures.
73 0386 1 The data accumulated per function (and also broken out for significant
74 0387 1 subfunctions) includes the number of functions executed, the number of
75 0388 1 modifiers, the number of disk reads, disk writes, and cache reads, the
76 0389 1 amount of CPU time, and the number of page faults.
77 0390 1 -
78 0391 1
79 0392 1
80 0393 1 System space data array. Each measured parameter is contained in a vector
81 0394 1 with one entry per function.
82 0395 1
83 0396 1
84 0397 1 EXTERNAL
85 0398 1 PMSSGL_FCP : VECTOR ADDRESSING_MODE (ABSOLUTE);
86 0399 1 ! base of FCP measurement array
87 0400 1
88 0401 1 BIND
89 0402 1 PMSSGL_COUNT = PMSSGL_FCP + 0 : VECTOR [10],
90 0403 1 PMSSGL_MCNT = PMSSGL_FCP + 40 : VECTOR [10],
91 0404 1 PMSSGL_READ = PMSSGL_FCP + 80 : VECTOR [10],
92 0405 1 PMSSGL_WRITE = PMSSGL_FCP + 120 : VECTOR [10],
93 0406 1 PMSSGL_CACHE = PMSSGL_FCP + 160 : VECTOR [10],
94 0407 1 PMSSGL_CPU = PMSSGL_FCP + 200 : VECTOR [10],
95 0408 1 PMSSGL_PFA = PMSSGL_FCP + 240 : VECTOR [10];
96 0409 1
97 0410 1
98 0411 1 Running totals maintained by FCP routines.
99 0412 1
100 0413 1
101 0414 1 GLOBAL
102 0415 1 PMS_TOT_READ, ! total disk reads
103 0416 1 PMS_TOT_WRITE, ! total disk writes
104 0417 1 PMS_TOT_CACHE; ! total cache reads
105 0418 1
106 0419 1
107 0420 1 Base values of parameters at start of this function.
108 0421 1
109 0422 1
110 0423 1 OWN
111 0424 1 PMS_FNC_READ,
112 0425 1 PMS_FNC_WRITE,
113 0426 1 PMS_FNC_CACHE,
114 0427 1 PMS_FNC_CPU,
115 0428 1 PMS_FNC_PFA;
116 0429 1
117 0430 1
118 0431 1 Base values of parameters at start of current subfunction.
119 0432 1
120 0433 1
121 0434 1
122 0435 1 GLOBAL
123 0436 1 PMS_SUB_NEST; ! nested subfunction flag
124 0437 1
```

PMS
V04-000

:	125	0438	1	OWN
:	126	0439	1	
:	127	0440	1	
:	128	0441	1	
:	129	0442	1	
:	130	0443	1	
:	131	0444	1	

PMS_SUB_FUNC,
PMS_SUB_READ,
PMS_SUB_WRITE,
PMS_SUB_CACHE,
PMS_SUB_CPU,
PMS_SUB_PFA;

L 6
16-Sep-1984 01:12:50
14-Sep-1984 12:29:47

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[F11A.SRC]PMS.B32;1

Page 4
(2)

! subfunction code

RDI
VOI


```
133 0445 1 GLOBAL ROUTINE PMS_START : NOVALUE =
134 0446 1
135 0447 1 ++
136 0448 1
137 0449 1 FUNCTIONAL DESCRIPTION:
138 0450 1
139 0451 1 This routine initiates measurement for the main function being executed.
140 0452 1
141 0453 1
142 0454 1 CALLING SEQUENCE:
143 0455 1 PMS_START ()
144 0456 1
145 0457 1 INPUT PARAMETERS:
146 0458 1 NONE
147 0459 1
148 0460 1 IMPLICIT INPUTS:
149 0461 1 NONE
150 0462 1
151 0463 1 OUTPUT PARAMETERS:
152 0464 1 NONE
153 0465 1
154 0466 1 IMPLICIT OUTPUTS:
155 0467 1 NONE
156 0468 1
157 0469 1 ROUTINE VALUE:
158 0470 1 NONE
159 0471 1
160 0472 1 SIDE EFFECTS:
161 0473 1 NONE
162 0474 1
163 0475 1 --
164 0476 1
165 0477 2 BEGIN
166 0478 2
167 0479 2 LOCAL
168 0480 2 PROCESS_HEADER : REF BBLOCK; ! pointer to FCP process header
169 0481 2
170 0482 2 EXTERNAL
171 0483 2 CTL$GL_PHD : REF BBLOCK ADDRESSING_MODE (ABSOLUTE);
172 0484 2 ! address of process header in control region
173 0485 2
174 0486 2
175 0487 2 ! To initialize measurement, we take copies of the running totals of all the
176 0488 2 ! parameters and stash them, so we can later compute the incremental usage.
177 0489 2 !
178 0490 2
179 0491 2 PROCESS_HEADER = .CTL$GL_PHD; ! get address of own process header
180 0492 2
181 0493 2 PMS_FNC_READ = .PMS_TOT_READ;
182 0494 2 PMS_FNC_WRITE = .PMS_TOT_WRITE;
183 0495 2 PMS_FNC_CACHE = .PMS_TOT_CACHE;
184 0496 2 PMS_FNC_CPU = .PROCESS_HEADER[PHD$CPUTIM];
185 0497 2 PMS_FNC_PFA = .PROCESS_HEADER[PHD$PAGEFLTS];
186 0498 2
187 0499 1 END; ! end of routine PMS_START
```

.TITLE PMS
.IDENT \V04-000\

.PSECT \$LOCKEDD1\$,NOEXE,2

00000 PMS_TOT_READ::
 .BLKB 4
00004 PMS_TOT_WRITE::
 .BLKB 4
00008 PMS_TOT_CACHE::
 .BLKB 4
0000C PMS_FNC_READ:
 .BLKB 4
00010 PMS_FNC_WRITE:
 .BLKB 4
00014 PMS_FNC_CACHE:
 .BLKB 4
00018 PMS_FNC_CPU:
 .BLKB 4
0001C PMS_FNC_PFA:
 .BLKB 4
00020 PMS_SUB_NEST::
 .BLKB 4
00024 PMS_SUB_FUNC:
 .BLKB 4
00028 PMS_SUB_READ:
 .BLKB 4
0002C PMS_SUB_WRITE:
 .BLKB 4
00030 PMS_SUB_CACHE:
 .BLKB 4
00034 PMS_SUB_CPU:
 .BLKB 4
00038 PMS_SUB_PFA:
 .BLKB 4

.EXTRN PMS\$GL_FCP, CTL\$GL_PHD

.PSECT \$CODE\$,NOWRT,2

	52	0000'	CF	9E	00002
	50	00000000G	9F	D0	00007
0C	A2		62	7D	0000E
14	A2	08	A2	D0	00012
18	A2	38	A0	D0	00017
1C	A2	4C	A0	D0	0001C
			04		00021

.ENTRY	PMS_START, Save R2	:	0445
MOVAB	PMS_TOT_READ, R2	:	
MOVL	@#CTL\$GL_PHD, PROCESS_HEADER	:	0491
MOVQ	PMS_TOT_READ, PMS_FNC_READ	:	0493
MOVL	PMS_TOT_CACHE, PMS_FNC_CACHE	:	0495
MOVL	56(PROCESS_HEADER), PMS_FNC_CPU	:	0496
MOVL	76(PROCESS_HEADER), PMS_FNC_PFA	:	0497
RET		:	0499

; Routine Size: 34 bytes, Routine Base: \$CODE\$ + 0000


```
189 0500 1 GLOBAL ROUTINE PMS_END : NOVALUE =
190 0501 1
191 0502 1 ++
192 0503 1
193 0504 1 FUNCTIONAL DESCRIPTION:
194 0505 1
195 0506 1 This routine ends measurement for the current main function. It
196 0507 1 subtracts the stored base values from the running totals of the
197 0508 1 parameters and accumulates the delta in the system space cells
198 0509 1 for the particular function.
199 0510 1
200 0511 1
201 0512 1 CALLING SEQUENCE:
202 0513 1 PMS_END ()
203 0514 1
204 0515 1 INPUT PARAMETERS:
205 0516 1 NONE
206 0517 1
207 0518 1 IMPLICIT INPUTS:
208 0519 1 IO_PACKET: address of I/O packet of this function
209 0520 1
210 0521 1 OUTPUT PARAMETERS:
211 0522 1 NONE
212 0523 1
213 0524 1 IMPLICIT OUTPUTS:
214 0525 1 NONE
215 0526 1
216 0527 1 ROUTINE VALUE:
217 0528 1 NONE
218 0529 1
219 0530 1 SIDE EFFECTS:
220 0531 1 measurement data base updated
221 0532 1
222 0533 1 --
223 0534 1
224 0535 2 BEGIN
225 0536 2
226 0537 2 BIND
227 0538 2 FUNCTAB = UPLIT BYTE ! table to translate function to array index
228 0539 2 ! IOS_ACCESS,
229 0540 2 ! IOS_CREATE,
230 0541 2 ! IOS_DEACCESS,
231 0542 2 ! IOS_DELETE,
232 0543 2 ! IOS_MODIFY,
233 0544 2 ! IOS_ACPCONTROL)
234 0545 2 : VECTOR [,BYTE];
235 0546 2
236 0547 2 LOCAL
237 0548 2 D, ! value of parameter change
238 0549 2 FUNCTION : BBLOCK[4], ! I/O function code, including modifiers
239 0550 2 J, ! array index
240 0551 2 PROCESS_HEADER : REF BBLOCK; ! address of own process header
241 0552 2
242 0553 2 EXTERNAL
243 0554 2 IO_PACKET : REF BBLOCK, ! address of I/O packet
244 0555 2 CT$GL_PHD : REF BBLOCK ADDRESSING_MODE (ABSOLUTE);
245 0556 2 ! address of process header in control region
```



```
246 0557 2 ! If there is a subfunction open, close it first.
247 0558 2 !
248 0559 2 !
249 0560 2 IF .PMS_SUB_NEST NEQ 0
250 0561 2 THEN
251 0562 2 BEGIN
252 0563 2 PMS_SUB_NEST = 1;
253 0564 2 PMS_END_SUB ();
254 0565 2 END;
255 0566 2 !
256 0567 2 ! Derive the table index from the function code by searching the function
257 0568 2 ! code table. If the code is not found, we do not record data.
258 0569 2 !
259 0570 2 !
260 0571 2 J =
261 0572 2 BEGIN
262 0573 2 INCR I FROM 0 TO 5 DO
263 0574 2 IF .IO_PACKET[IRPSV_FCODE] EQL .FUNCTAB[I]
264 0575 2 THEN EXITLOOP .I
265 0576 2 END;
266 0577 2 !
267 0578 2 IF .J EQL -1 THEN RETURN;
268 0579 2 !
269 0580 2 ! Compute the deltas and accumulate them into the system space array.
270 0581 2 ! Note that we explicitly compute the change for each parameter and then
271 0582 2 ! add it into the data base cell, to prevent windows in which the value of
272 0583 2 ! the parameter is held in a local. This is because we cannot count on the
273 0584 2 ! compiler to generate simple stores which would be hazard free.
274 0585 2 !
275 0586 2 !
276 0587 2 PROCESS HEADER = .CTL$GL PHD;
277 0588 2 FUNCTION = .IO_PACKET[IRPSW_FUNC];
278 0589 2 !
279 0590 2 PMSSGL COUNT[J] = .PMSSGL_COUNT[J] + 1;
280 0591 2 IF .FUNCTION[IOSV_ACCESS]
281 0592 2 THEN PMSSGL MCNT[J] = .PMSSGL_MCNT[J] + 1;
282 0593 2 IF .FUNCTION[IOSV_CREATE]
283 0594 2 THEN PMSSGL MCNT[J] = .PMSSGL_MCNT[J] + 1;
284 0595 2 IF .FUNCTION[IOSV_DELETE]
285 0596 2 THEN PMSSGL MCNT[J] = .PMSSGL_MCNT[J] + 1;
286 0597 2 D = .PMS TOT READ - .PMS_FNC READ;
287 0598 2 PMSSGL READ[J] = .PMSSGL_READ[J] + .D;
288 0599 2 D = .PMS TOT WRITE - .PMS_FNC WRITE;
289 0600 2 PMSSGL WRITE[J] = .PMSSGL_WRITE[J] + .D;
290 0601 2 D = .PMS TOT CACHE - .PMS_FNC CACHE;
291 0602 2 PMSSGL CACHE[J] = .PMSSGL_CACHE[J] + .D;
292 0603 2 D = .PROCESS HEADER[PHD$L_CPUTIM] - .PMS_FNC_CPU;
293 0604 2 PMSSGL CPU[J] = .PMSSGL_CPU[J] + .D;
294 0605 2 D = .PROCESS HEADER[PHD$L_PAGEFLTS] - .PMS_FNC_PFA;
295 0606 2 PMSSGL_PFA[J] = .PMSSGL_PFA[J] + .D;
296 0607 2 !
297 0608 1 END; ! end of routine PMS_END
```

38 36 35 34 33 32 00022 P.AAA: .BYTE 50, 51, 52, 53, 54, 56 ;

				FUNCTAB=	P.AAA		
				.EXTRN	IO_PACKET		
				001C 00000	.ENTRY	PMS_END, Save R2,R3,R4	0500
		54	0000'	CF 9E 00002	MOVAB	PMS_SUB_NEST, R4	
		53	00000000G	9F 9E 00007	MOVAB	@#PMS\$GL MCNT, R3	
				64 D5 0000E	TSTL	PMS_SUB_NEST	0560
				08 13 00010	BEQL	1	
		64		01 D0 00012	MOVL	#1, PMS_SUB_NEST	0563
		CF	0000V	00 FB 00015	CALLS	#0, PMS_END-SUB	0564
		51	0000G	CF D0 0001A	MOVL	IO_PACKET, R1	0574
				50 D4 0001F	CLRL	I	
		52	D5 AF40	9A 00021	MOVZBL	FUNCTAB[I], R2	
52	20	A1	06	00 ED 00026	CMPZV	#0, #6, 32(R1), R2	
		EF		07 13 0002C	BEQL	3	
				05 F3 0002E	AOBLEQ	#5, I, 2	
		50		01 CE 00032	MNEGL	#1, J	0573
		FFFFFFFFFF	8F	50 D1 00035	CMPL	J, #-1	0578
				5D 13 0003C	BEQL	7	
		52	00000000G	9F D0 0003E	MOVL	@#CTL\$GL PHD, PROCESS_HEADER	0587
		51	20 A1	3C 00045	MOVZWL	32(R1), FUNCTION	0588
			D8 A340	D6 00049	INCL	PMS\$GL COUNT[J]	0590
		03	51	06 E1 0004D	BBC	#6, FUNCTION, 4	0591
				6340 D6 00051	INCL	PMS\$GL MCNT[J]	0592
				51 95 00054	TSTB	FUNCTION	0593
				03 18 00056	BGEQ	5	
		03	51	6340 D6 00058	INCL	PMS\$GL MCNT[J]	0594
				08 E1 0005B	BBC	#8, FUNCTION, 6	0595
				6340 D6 0005F	INCL	PMS\$GL MCNT[J]	0596
		51	E0 A4	EC A4 C3 00062	SUBL3	PMS_FNC_READ, PMS_TOT_READ, D	0597
			28 A340	51 C0 00068	ADDL2	D, PMS\$GL_READ[J]	0598
		51	E4 A4	F0 A4 C3 0006D	SUBL3	PMS_FNC_WRITE, PMS_TOT_WRITE, D	0599
			50 A340	51 C0 00073	ADDL2	D, PMS\$GL_WRITE[J]	0600
		51	E8 A4	F4 A4 C3 00078	SUBL3	PMS_FNC_CACHE, PMS_TOT_CACHE, D	0601
			78 A340	51 C0 0007E	ADDL2	D, PMS\$GL_CACHE[J]	0602
		51	38 A2	F8 A4 C3 00083	SUBL3	PMS_FNC_CPU, 56(PROCESS_HEADER), D	0603
			00A0 C340	51 C0 00089	ADDL2	D, PMS\$GL_CPU[J]	0604
		51	4C A2	FC A4 C3 0008F	SUBL3	PMS_FNC_PFA, 76(PROCESS_HEADER), D	0605
			00C8 C340	51 C0 00095	ADDL2	D, PMS\$GL_PFA[J]	0606
				04 0009B	RET		0608

; Routine Size: 156 bytes, Routine Base: \$CODE\$ + 0028

```
299 0609 1 GLOBAL ROUTINE PMS_START_SUB (INDEX) : NOVALUE =
300 0610 1
301 0611 1 ++
302 0612 1
303 0613 1 FUNCTIONAL DESCRIPTION:
304 0614 1
305 0615 1 This routine starts metering for the indicated subfunction.
306 0616 1
307 0617 1
308 0618 1 CALLING SEQUENCE:
309 0619 1 PMS_START_SUB (ARG1)
310 0620 1
311 0621 1 INPUT PARAMETERS:
312 0622 1 ARG1: index of measurement array to use
313 0623 1
314 0624 1 IMPLICIT INPUTS:
315 0625 1 NONE
316 0626 1
317 0627 1 OUTPUT PARAMETERS:
318 0628 1 NONE
319 0629 1
320 0630 1 IMPLICIT OUTPUTS:
321 0631 1 NONE
322 0632 1
323 0633 1 ROUTINE VALUE:
324 0634 1 NONE
325 0635 1
326 0636 1 SIDE EFFECTS:
327 0637 1 NONE
328 0638 1
329 0639 1 --
330 0640 1
331 0641 2 BEGIN
332 0642 2
333 0643 2 LOCAL
334 0644 2 PROCESS_HEADER : REF BBLOCK; ! pointer to FCP process header
335 0645 2
336 0646 2 EXTERNAL
337 0647 2 CTL$GL_PHD : REF BBLOCK ADDRESSING_MODE (ABSOLUTE);
338 0648 2 ! address of process header in control region
339 0649 2
340 0650 2 ! We copy the current running totals into subfunction holding cells to
341 0651 2 ! compute the deltas later. Note that since the extend subfunction can be
342 0652 2 ! reentered, we do nothing if the depth count is already non-zero.
343 0653 2
344 0654 2
345 0655 2 PMS_SUB_NEST = .PMS_SUB_NEST + 1;
346 0656 2 IF .PMS_SUB_NEST NEQ 1 THEN RETURN;
347 0657 2 PROCESS_HEADER = .CTL$GL_PHD;
348 0658 2
349 0659 2 PMS_SUB_FUNC = .INDEX;
350 0660 2 PMS_SUB_READ = .PMS_TOT_READ;
351 0661 2 PMS_SUB_WRITE = .PMS_TOT_WRITE;
352 0662 2 PMS_SUB_CACHE = .PMS_TOT_CACHE;
353 0663 2 PMS_SUB_CPU = .PROCESS_HEADER[PHD$$_CPU];
354 0664 2 PMS_SUB_PFA = .PROCESS_HEADER[PHD$$_PAGEFLTS];
355 0665 2
```


PMS
V04-000

: 356

0666 1 END;

F 7
16-Sep-1984 01:12:50
14-Sep-1984 12:29:47

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[F11A.SRC]PMS.B32;1

Page 11
(5)

! end of routine PMS_START_SUB

				0004 00000	.ENTRY	PMS_START_SUB, Save R2	:	0609
	52	0000'	CF	9E 00002	MOVAB	PMS_SUB_NEST, R2	:	
			62	D6 00007	INCL	PMS_SUB_NEST	:	0655
	01		62	D1 00009	CMPL	PMS_SUB_NEST, #1	:	0656
			20	12 0000C	BNEQ	1\$	:	
	50	00000000G	9F	D0 0000E	MOVL	@#CTL\$GL PHD, PROCESS_HEADER	:	0657
04	A2	04	AC	D0 00015	MOVL	INDEX, PMS_SUB_FUNC	:	0659
08	A2	E0	A2	7D 0001A	MOVQ	PMS_TOT_READ, PMS_SUB_READ	:	0660
10	A2	E8	A2	D0 0001F	MOVL	PMS_TOT_CACHE, PMS_SUB_CACHE	:	0662
14	A2	38	A0	D0 00024	MOVL	56(PROCESS_HEADER), PMS_SUB_CPU	:	0663
18	A2	4C	A0	D0 00029	MOVL	76(PROCESS_HEADER), PMS_SUB_PFA	:	0664
			04	0002E 1\$:	RET		:	0666

; Routine Size: 47 bytes, Routine Base: \$CODE\$ + 00C4

```
0667 1 GLOBAL ROUTINE PMS_END_SUB : NOVALUE =
0668 1
0669 1 ++
0670 1
0671 1 FUNCTIONAL DESCRIPTION:
0672 1
0673 1     This routine ends metering for the currently active subfunction.
0674 1
0675 1
0676 1 CALLING SEQUENCE:
0677 1     PMS_END_SUB ()
0678 1
0679 1 INPUT PARAMETERS:
0680 1     NONE
0681 1
0682 1 IMPLICIT INPUTS:
0683 1     NONE
0684 1
0685 1 OUTPUT PARAMETERS:
0686 1     NONE
0687 1
0688 1 IMPLICIT OUTPUTS:
0689 1     NONE
0690 1
0691 1 ROUTINE VALUE:
0692 1     NONE
0693 1
0694 1 SIDE EFFECTS:
0695 1     measurement data base updated
0696 1
0697 1 --
0698 1
0699 2 BEGIN
0700 2
0701 2 LOCAL
0702 2     J,                ! array index
0703 2     D,                ! parameter difference
0704 2     PROCESS_HEADER : REF BBLOCK; ! pointer to FCP process header
0705 2
0706 2 EXTERNAL
0707 2     CTL$GL_PHD      : REF BBLOCK ADDRESSING_MODE (ABSOLUTE);
0708 2                     ! address of process header in control region
0709 2
0710 2 ! Decrement the nesting count. If non-zero, we are in a nested extend and
0711 2 ! do nothing.
0712 2
0713 2
0714 2 PMS_SUB_NEST = .PMS_SUB_NEST - 1;
0715 2 IF .PMS_SUB_NEST NEQ 0 THEN RETURN;
0716 2
0717 2 ! Now compute the delta for each parameter by subtracting the base from the
0718 2 ! running total. Record it by adding into the system array. Also deduct
0719 2 ! the delta from the charge for the main function by adding it into the
0720 2 ! main function base.
0721 2
0722 2
0723 2 PROCESS_HEADER = .CTL$GL_PHD;
```



```

: 415      0724 2 J = .PMS_SUB_FUNC;                ! get array index
: 416      0725 2
: 417      0726 2 PMSS$GL_COUNT[J] = .PMSS$GL_COUNT[J] + 1;
: 418      0727 2
: 419      0728 2 D = .PMS_TOT_READ - .PMS_SUB_READ;
: 420      0729 2 PMSS$GL_READ[J] = .PMSS$GL_READ[J] + .D;
: 421      0730 2 PMS_FNC_READ = .PMS_FNC_READ + .D;
: 422      0731 2
: 423      0732 2 D = .PMS_TOT_WRITE - .PMS_SUB_WRITE;
: 424      0733 2 PMSS$GL_WRITE[J] = .PMSS$GL_WRITE[J] + .D;
: 425      0734 2 PMS_FNC_WRITE = .PMS_FNC_WRITE + .D;
: 426      0735 2
: 427      0736 2 D = .PMS_TOT_CACHE - .PMS_SUB_CACHE;
: 428      0737 2 PMSS$GL_CACHE[J] = .PMSS$GL_CACHE[J] + .D;
: 429      0738 2 PMS_FNC_CACHE = .PMS_FNC_CACHE + .D;
: 430      0739 2
: 431      0740 2 D = .PROCESS_HEADER[PHD$&L_CPUTIM] - .PMS_SUB_CPU;
: 432      0741 2 PMSS$GL_CPU[J] = .PMSS$GL_CPU[J] + .D;
: 433      0742 2 PMS_FNC_CPU = .PMS_FNC_CPU + .D;
: 434      0743 2
: 435      0744 2 D = .PROCESS_HEADER[PHD$&L_PAGEFLTS] - .PMS_SUB_PFA;
: 436      0745 2 PMSS$GL_PFA[J] = .PMSS$GL_PFA[J] + .D;
: 437      0746 2 PMS_FNC_PFA = .PMS_FNC_PFA + .D;
: 438      0747 2
: 439      0748 1 END;                                ! end of routine PMS_END_SUB
```

				001C 00000	.ENTRY	PMS_END_SUB, Save R2,R3,R4	: 0667
	54	00000000G	9F	9E 00002	MOVAB	@#PMSS\$GL_COUNT, R4	
	53	0000'	CF	9E 00009	MOVAB	PMS_SUB_NEST, R3	
			63	D7 0000E	DECL	PMS_SUB_NEST	: 0714
			5C	12 00010	BNEQ	1\$	: 0715
	52	00000000G	9F	D0 00012	MOVL	@#CTL\$GL_PHD, PROCESS_HEADER	: 0723
	50	04	A3	D0 00019	MOVL	PMS_SUB_FUNC, J	: 0724
			6440	D6 0001D	INCL	PMSS\$GL_COUNT[J]	: 0726
51	E0	A3	08	A3 C3 00020	SUBL3	PMS_SUB_READ, PMS_TOT_READ, D	: 0728
	50	A440		51 C0 00026	ADDL2	D, PMSS\$GL_READ[J]	: 0729
	EC	A3		51 C0 0002B	ADDL2	D, PMS_FNC_READ	: 0730
51	E4	A3	0C	A3 C3 0002F	SUBL3	PMS_SUB_WRITE, PMS_TOT_WRITE, D	: 0732
	78	A440		51 C0 00035	ADDL2	D, PMSS\$GL_WRITE[J]	: 0733
	F0	A3		51 C0 0003A	ADDL2	D, PMS_FNC_WRITE	: 0734
51	E8	A3	10	A3 C3 0003E	SUBL3	PMS_SUB_CACHE, PMS_TOT_CACHE, D	: 0736
	00A0	C440		51 C0 00044	ADDL2	D, PMSS\$GL_CACHE[J]	: 0737
	F4	A3		51 C0 0004A	ADDL2	D, PMS_FNC_CACHE	: 0738
51	38	A2	14	A3 C3 0004E	SUBL3	PMS_SUB_CPU, 56(PROCESS_HEADER), D	: 0740
	00C8	C440		51 C0 00054	ADDL2	D, PMSS\$GL_CPU[J]	: 0741
	F8	A3		51 C0 0005A	ADDL2	D, PMS_FNC_CPU	: 0742
51	4C	A2	18	A3 C3 0005E	SUBL3	PMS_SUB_PFA, 76(PROCESS_HEADER), D	: 0744
	00F0	C440		51 C0 00064	ADDL2	D, PMSS\$GL_PFA[J]	: 0745
	FC	A3		51 C0 0006A	ADDL2	D, PMS_FNC_PFA	: 0746
				04 0006E 1\$:	RET		: 0748

; Routine Size: 111 bytes, Routine Base: \$CODE\$ + 00F3

: 440 0749 1
: 441 0750 1 END
: 442 0751 0 ELUDOM

PSECT SUMMARY

Name	Bytes	Attributes
\$LOCKEDD1\$	60 NOVEC, WRT, RD	NOEXE,NOSHR, LCL, REL, CON,NOPIC,ALIGN(2)
\$CODE\$	354 NOVEC,NOWRT, RD	EXE,NOSHR, LCL, REL, CON,NOPIC,ALIGN(2)

Library Statistics

File	----- Total	Symbols Loaded	----- Percent	Pages Mapped	Processing Time
_\$255\$DUA28:[SYSLIB]LIB.L32;1	18619	14	0	1000	00:01.9

COMMAND QUALIFIERS

: BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS\$:PMS/OBJ=OBJ\$:PMS MSRC\$:PMS/UPDATE=(ENH\$:PMS)

: Size: 348 code + 66 data bytes
: Run Time: 00:12.2
: Elapsed Time: 00:30.7
: Lines/CPU Min: 3696
: Lexemes/CPU-Min: 15283
: Memory Used: 118 pages
: Compilation Complete

0166 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

MODIFY
LIS

REQUEL
LIS

RWATTR
LIS

SCHFCB
LIS

MAKACC
LIS

MPWIND
LIS

MAPUBN
LIS

PMS
LIS

RDHEDR
LIS

RWUB
LIS

RETDIR
LIS

ROBLOK
LIS

SMALOC
LIS

MOUNT
LIS

MAKMB
LIS

MAKSTR
LIS

MAKTHOR
LIS